SOA @ T-Mobile –
Vollautomatische Service Provisionierung auf dem ESB

W-JAX 2008, München, Carsten Sensler & Andre Karalus

Agenda

- Prerequisites
- SOA Backplane in a nutshell
- Design time – in detail
- Runtime – in detail
- Provisioning time – in detail
- Organizational aspects – international focus
- Summary

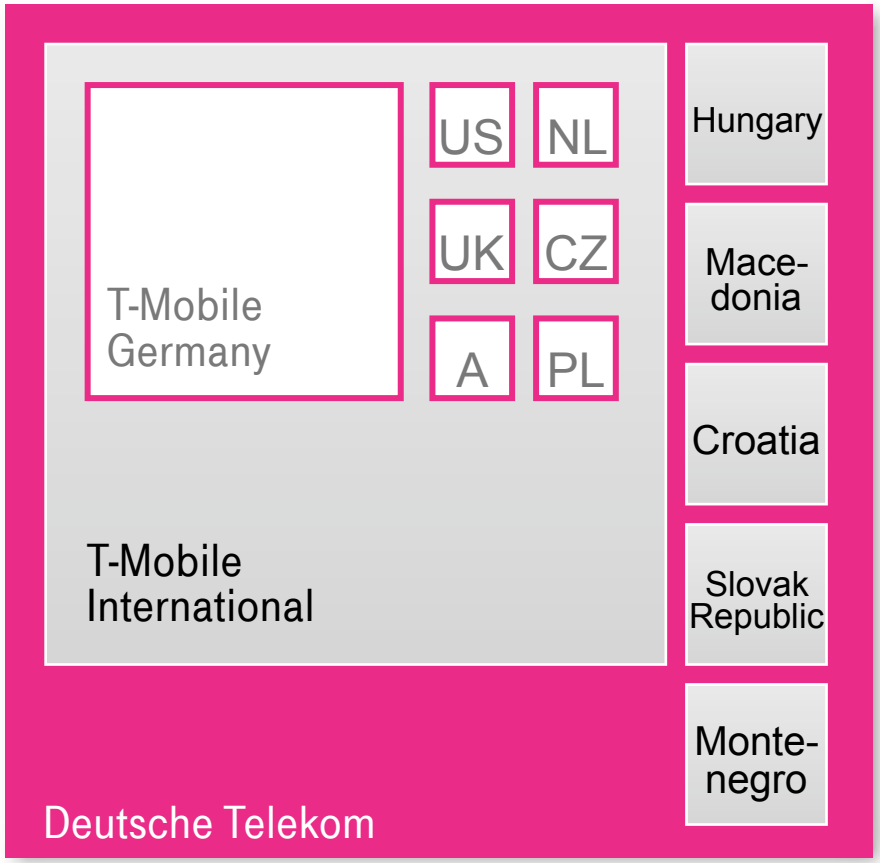
Who we are?

- Dipl.-Ing. Carsten Sensler
  - Employee of T-Mobile Deutschland GmbH since April 2007 (but since December 2005 working in the SOA Backplane program)
  - Department of Enterprise Integration
  - System & Solution Designer
  - Functional leader of the international Service Provisioning Team
  - http://www.sensler.de

- Dipl.-Inf. Andre Karalus
  - Freelancer,
  - since March 2006 consultant for T-Mobile
  - Designer and developer of the runtime core component of the ESB in SOA Backplane and of the Core from the Service Repository

Prerequisites

Corporate Structure.
Subsidiaries and affiliates.

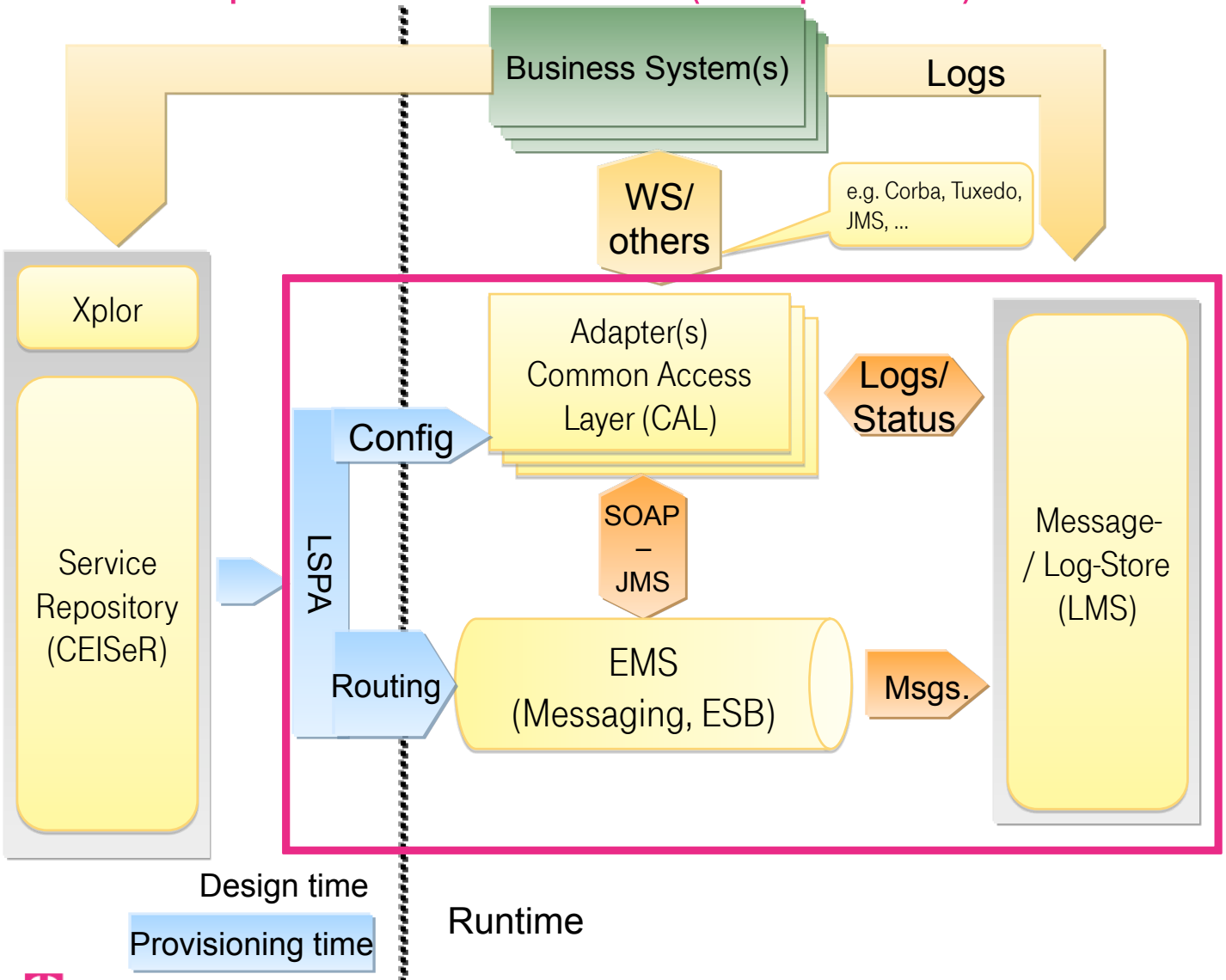


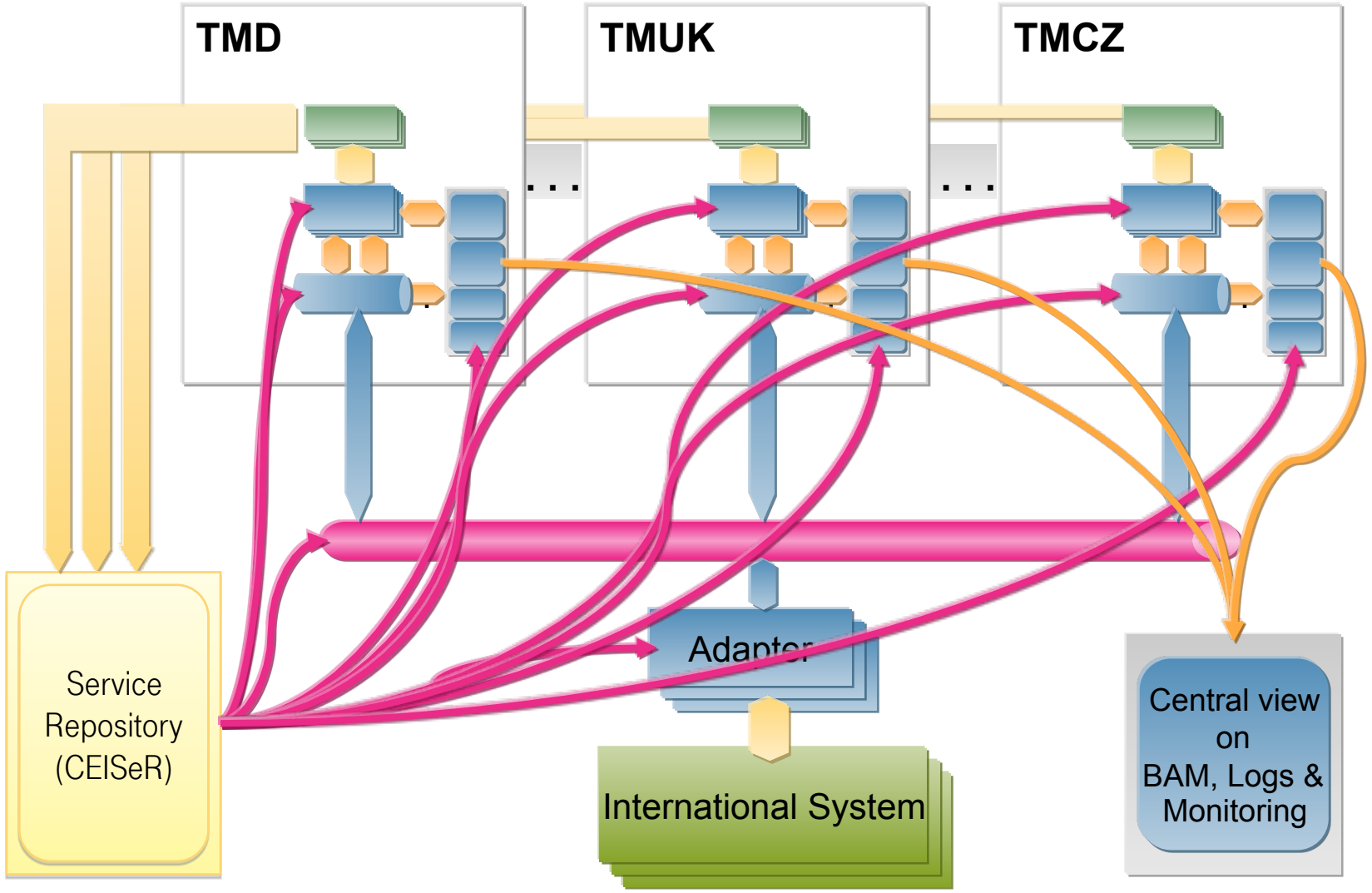
- Direct or indirect investments by Telekom Group in companies dealing with mobile communications in twelve countries
- The T-Mobile brand is represented in Germany, Austria, Hungary, Great Britain, the Czech Republic, the Netherlands, the Slovak Republic, Croatia and the USA
- Almost 125 million customers in the majority holdings

SOA Backplane Participant

SOA Backplane in nutshell

SOA Backplane – overview (simplified)




TMD
TMUK
TMCZ
Service Repository (CEISeR)
Adapter
International System
Central view on BAM, Logs & Monitoring

Intention of the SOA Backplane program

- SOA Backplane will deliver a number of software systems and standards, namely
  - a service bus which is the SOA communication infrastructure with static routing
    - Service repository
    - Access layer framework
    - Basic messaging infrastructure (JMS)
  - additional value adding components and functionality including
    - logging, monitoring
    - service contract management
    - business activity monitoring
    - transport components for B2B communication
  - the Backplane Guide and SOA Governance as a set of guidelines and rules as to how SOA will be implemented within T-Mobile.

Benefits of SOA BP

- Standardized architecture and interfaces and communication protocols
  - (e.g. technology independence, single interface type)
- Loosely-coupled systems
  - (e.g. releases and downtimes decoupled)
- Reuse

- Advanced logging and monitoring
  - (e.g. locate incidents in 1 step)
- Location independency
  - (e.g. individual IP addresses or firewall clearences do not matter)

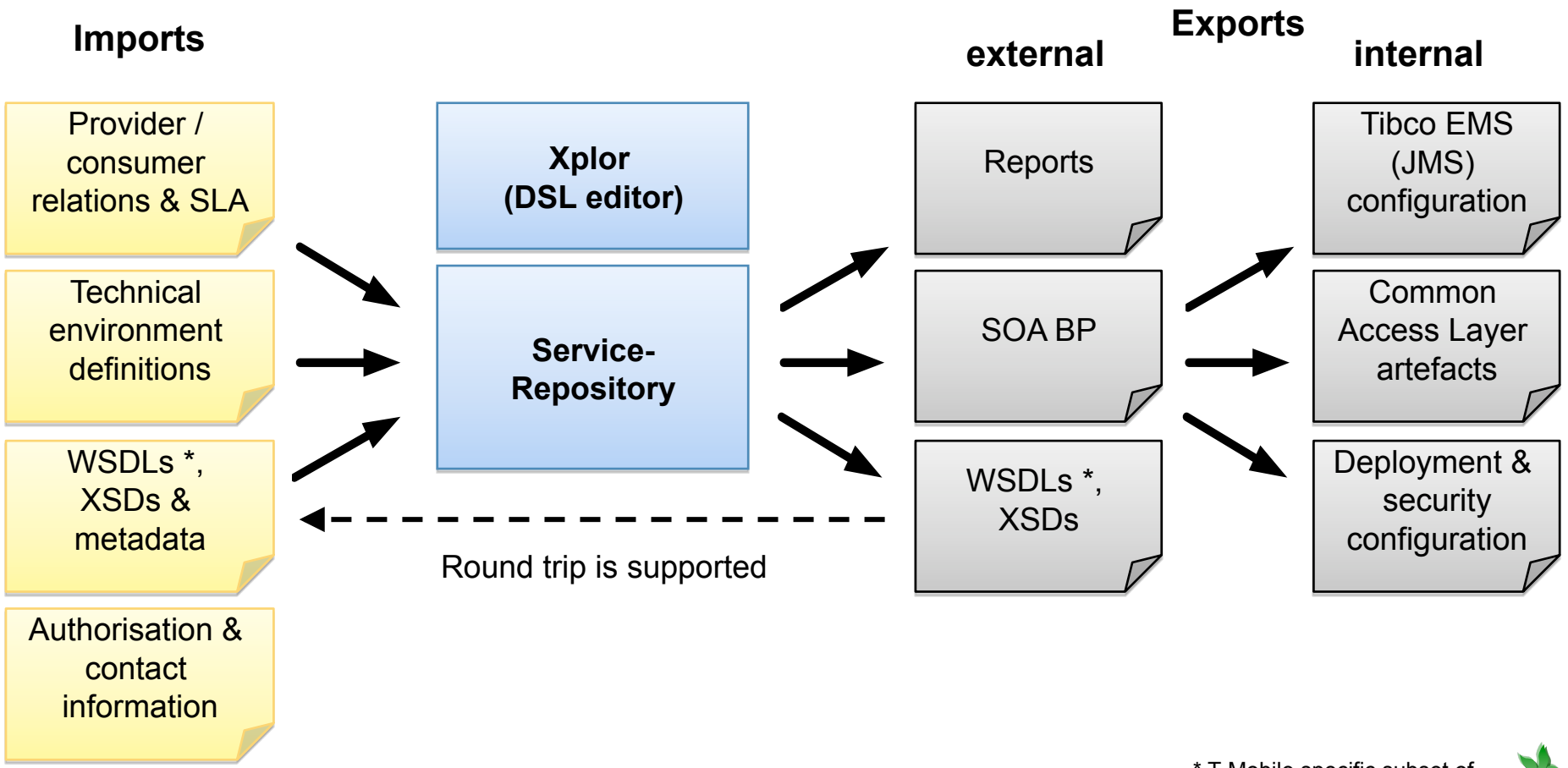
- This leads to
  - shorter Time-to-market
  - Cost savings

Static routing vs. dynamic routing

- The routing basically defines which Service Consumer (SC) is talking to which Service Provider (SP)
  - With dynamic routing the SC specifies the target within the (technical part of the) message. The ESB is then forwarding the message ad hoc to the SP
  - With static routing the SC identifies itself wanting to use a certain service. The ESB is configured to know the routing for that and then forward the message according to that configured information
- The advantage of dynamic routing is the flexibility to add new SC on the fly
- Downsides
  - No control who is using who or if a SP is used at all, no SC specific QoS enforceable
- The advantage of static routing is the full control over the ESB thus enabling SOA governance
- The downside is that all changes to routings have to be configured in the ESB before being used

SOA Backplane – Design time

SOA Backplane – Service Repository Interfaces

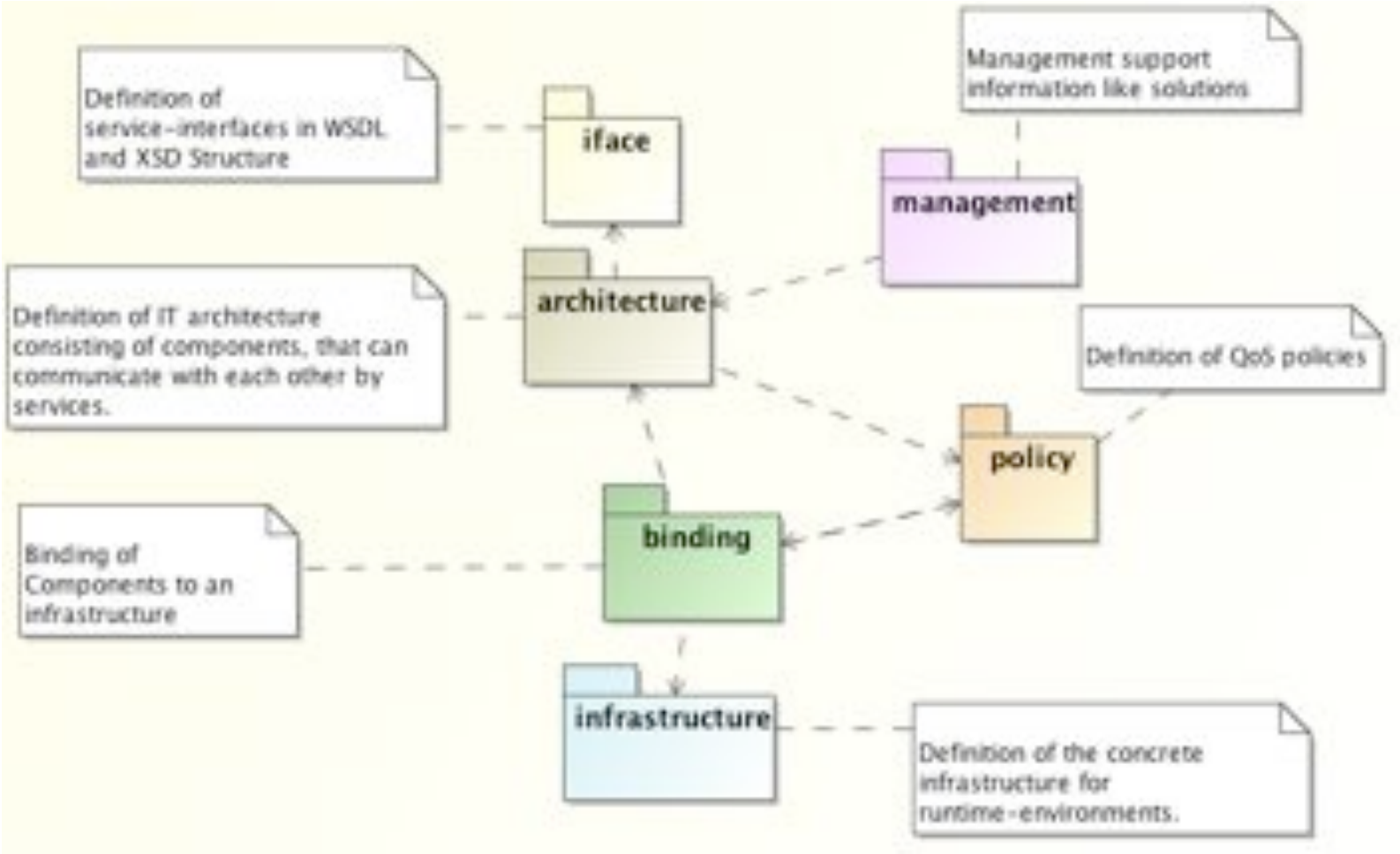


* T-Mobile specific subset of WSI basic profile 1.1

The Service Repository in a nutshell

- Provide all information needed by service participants for consistent service implementation and utilisation (architecture)
- Store definition of different SOA backplane environments and binding of service participants to these environments (binding)
- Support fully automated configuration of SOA backplane environments (dev, test, prod, ...) (service provisioning)
- Support SOA governance (service discovery, reuse) and impact analysis (change / incident contact information) (management)

The Underlying MetaModel



CEISeR - *.xadmin

- Specifies the following information
  - CEISeR-Namespaces
  - User of CEISeR
  - Access rights
  - Contact information
  - Concern (for regular notification of changes of a specific part of the CEISeR contents)

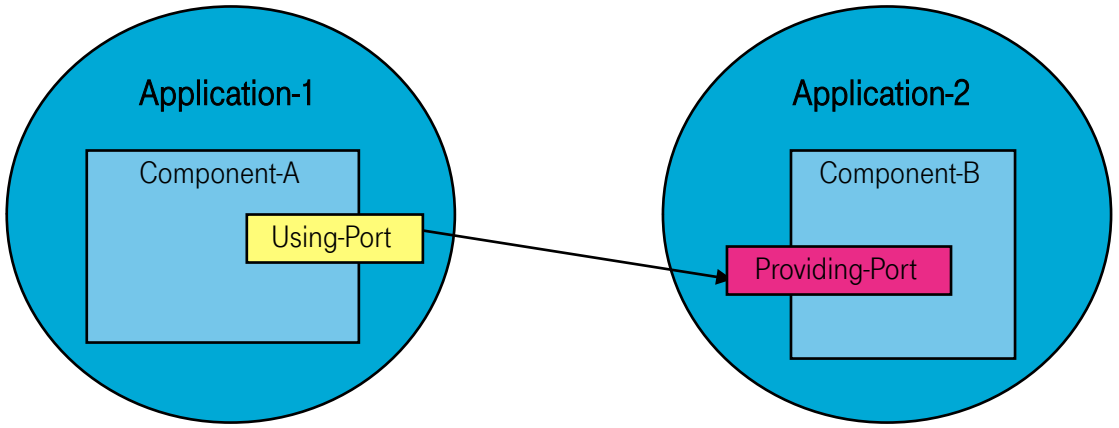
CEISeR – Service interface definition

- WSDLs and the included XSD
- All SOA Backplane compliant WSDL files have to include the EI XSD file (ei.messaging.datatypes.xsd)
- For designing a service interface definition for SOA Backplane there exist 23 rules (constraints)
  - wsdl-style MUST be "document/literal wrapped" (http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl/)
  - All operations MUST have a "SOABPException" – fault
  - wsdl:definitions MUST contain only ONE wsdl:portType-section
  - Enconding MUST be "UTF-8".

CEISeR – Architecture - *.xarchitecture

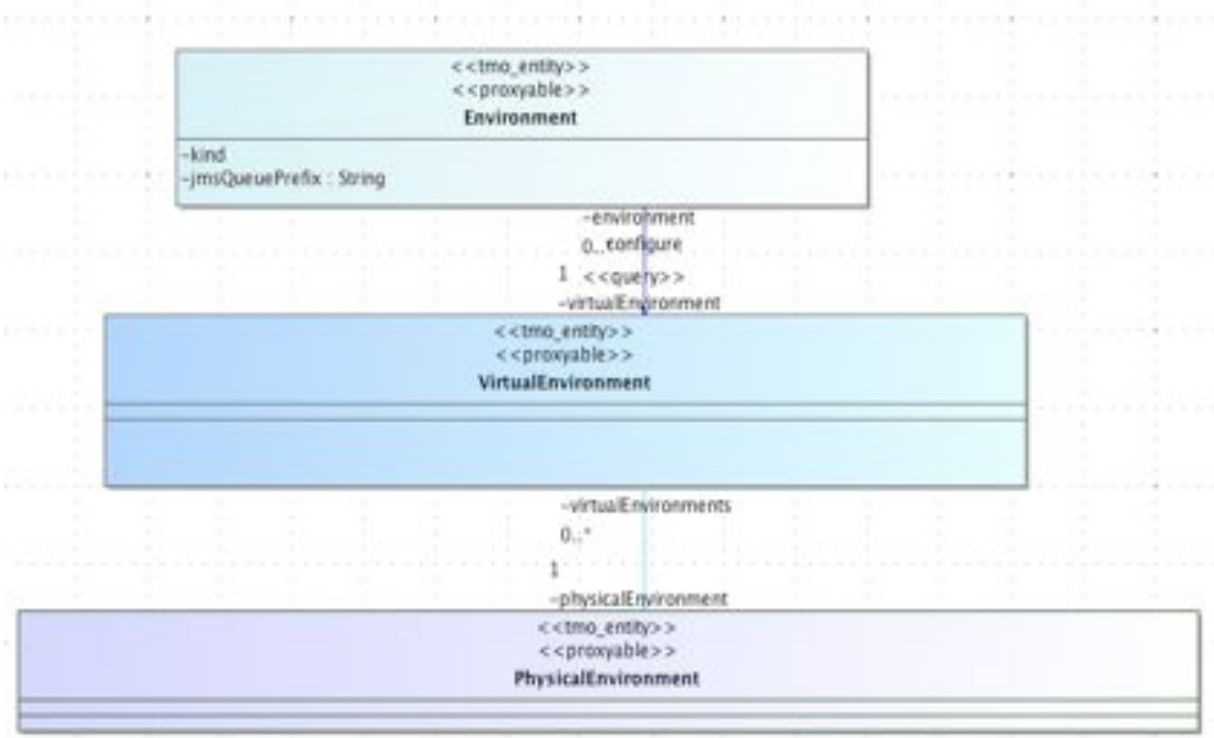
- The architecture in the context of CEISeR is defined by all
  - applications with their
  - components with their
  - ports and the
  - logical connection between the components

- The architecture reflects the enterprise architecture of the involved systems



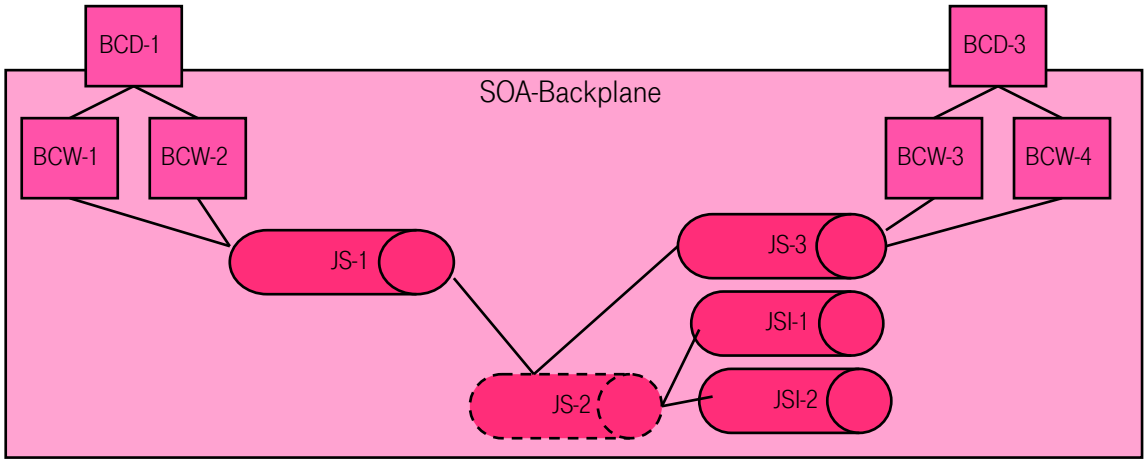
CEISeR – Infrastructure - *.xinfrastructure

- The infrastructure layer is separated into three layers
  - (SOABP Runtime) Environment
  - Virtual Environment
  - Physical environment



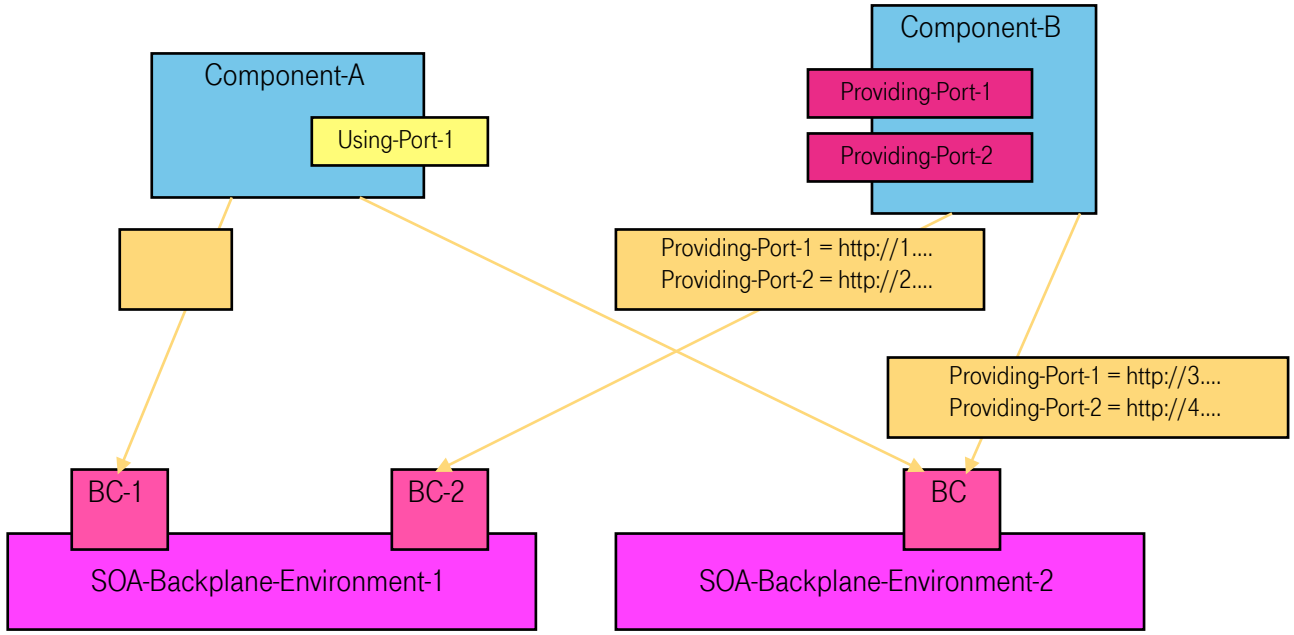
CEISeR – Infrastructure - *.xinfrastructure

- The internal architecture of the SOA-Backplane infrastructure consists of
  - **Binding-Components**: they can be load-balanced, so that a binding-component-dispatcher (BCD) dispatches the load to a set of binding-component-worker (BCW) (i.e. Common Access Layer – CAL)
  - **JMS-Servers**: they can be designed with more than one instances (JSI) for ensuring failover
- All direct or indirect connected JMS-Server and binding-components build a closed **environment**. There could be exist many environments. (i.e. Dev, T&A, Production). There is **no connection** between environments

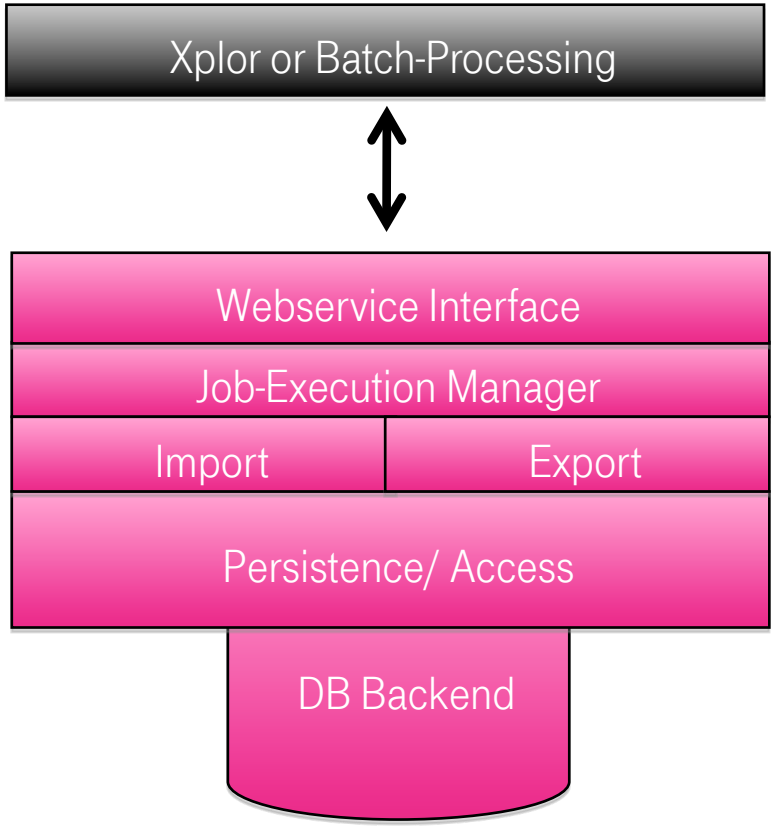


CEISeR – Binding - *.xbinding

- The architecture can be instantiated for eachexisting infrastructure-environment
- Each component has to be bound to exact one binding-component of each desired environment.
- For each providing-port, the URL where the binding-component can call the service-implementation must be published to the corresponding binding-component.



Technical design of CEISeR - simplified



CEISeR Metaphers

- Job Execution and Job description
  - Executes specified tasks in the job description file to CEISeR.
  - A job is executed via the job framework and the communication CEISeR-Server -> Client and vice versa is done via a web service
- Transaction number
  - Each “real” commit to the DB leads in an unique transaction number (similar to the revision concept of Subversion)
- Offline working with CEISeR
  - We work always offline with CEISeR
  - e.g: Synchronize your local model with the global CEISeR model via a job

How is CEISeR organized internally

- Every entity has got a namespace
  - A “directory”

- Every entity has got a name
- Namespaces are separated by dots
  - tmo.ei.icc.training

- namespace + . + name = CEISeR fully qualified name
  - tmo.ei.icc.training.Infrastructure

Control and audit changes

- Control access to the service repository
  - The access is subject to proper authentication
  - The account is managed in the company's central LDAP
- Restrict changes according to the organizational structure
  - The privileges to modify contents of the repository are subject to authorization
  - It's modeled with user and roles, roles have different privileges and relate to namespace hierarchies (every object in our repository belongs to a namespace)
- Audit what was changed, by whom and why
  - The set of changes applied to the repository belong to a transaction
  - The transaction can be reviewed later on, the changed objects, the user that did it and a description are recorded

Organize Artifacts

- It is not possible to name every single object when referring to the configuration - it becomes necessary to group single objects in order to have a larger granularity thus enabling manageability
  - Usually you have components, packages, modules, subsystems or the like

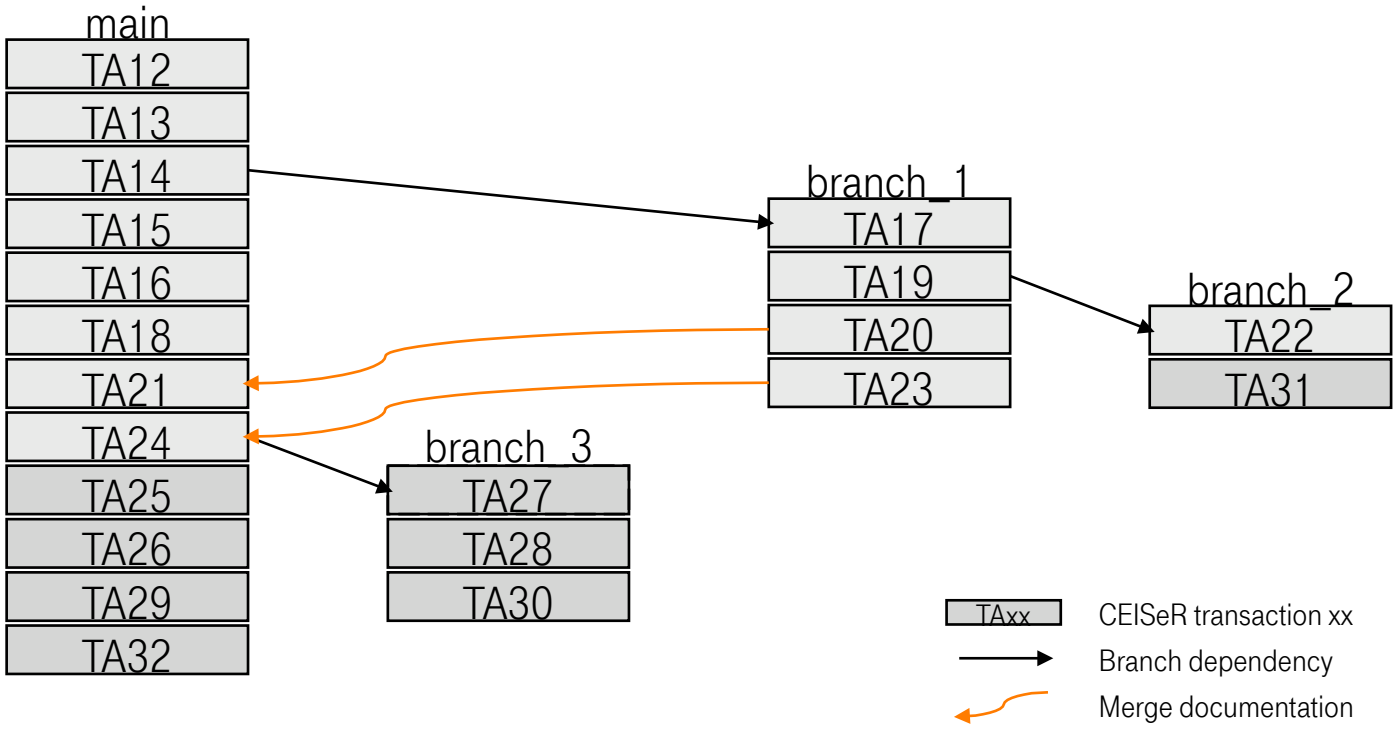
- The soa repository itself determines the structure by it's meta model
  - layers are interface, architecture, binding, infrastructure, policy, ...
  - Top level objects are used to refer to a logical subsystem
    - Application
    - BindingSet
    - Environment
    - ...

Create baselines at milestones and refer to them (1)

- Typical milestones are the begin of QA phase or the productive release
- We use the branching and tagging concept, this is a feature of the soa repository (i.e. the underlying core platform and it's build into the persistence layer)
- We are able to label a certain state of the repository and use it later on
  - To refer to it while browsing the repository
  - To build a service configuration on top of that state
  - To generate a report of that state

Create baselines at milestones and refer to them (2)

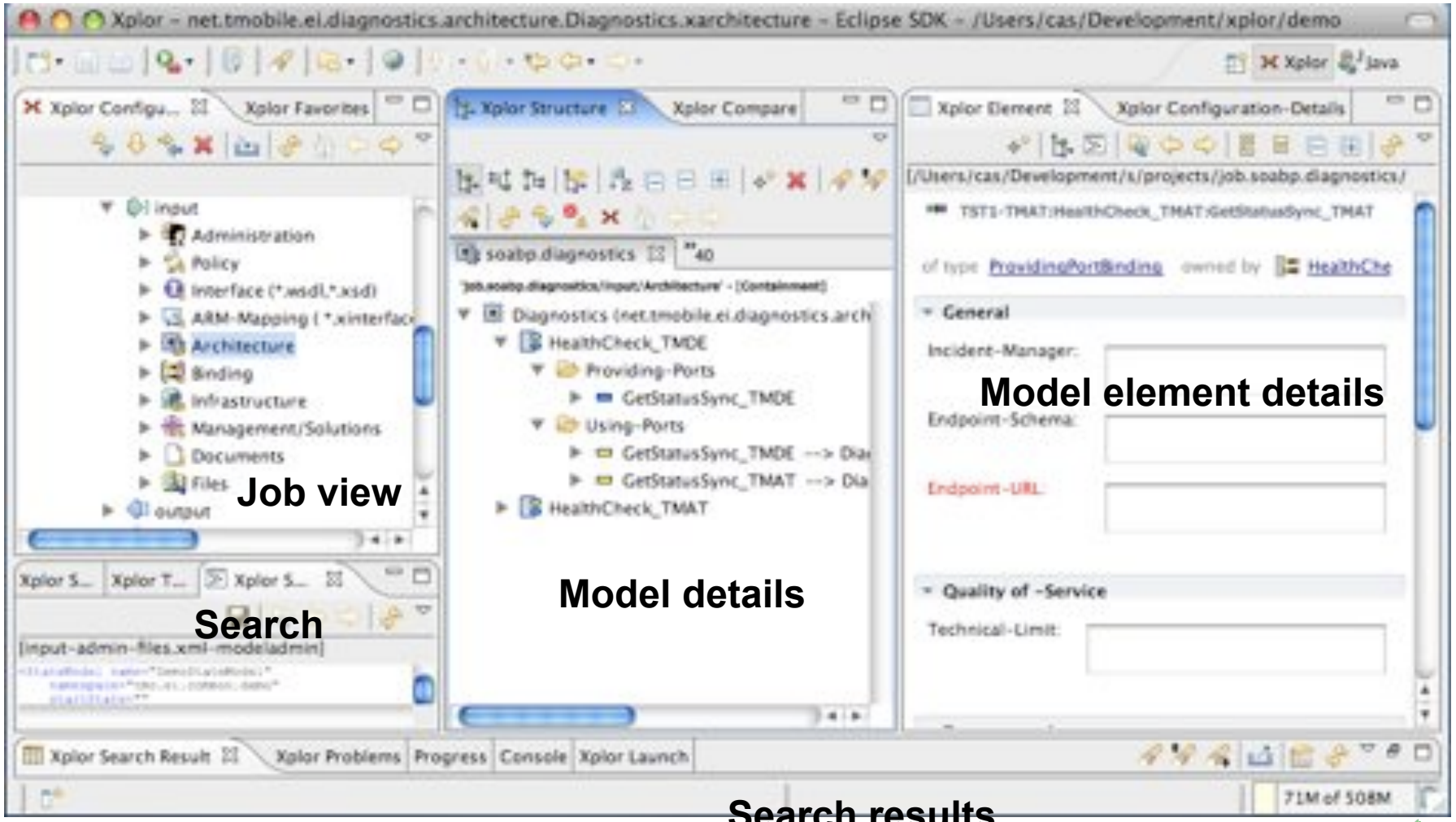
- It‘s possible to create a branch from a certain repository state
- This branches can be referred and fixes can be applied
- Changes can be merged back and this is documented



Xplor Eclipse based client

- Eclipse plug-in or Eclipse-based RCP
- DSL editor for CEISeR
  - Generic GUI based on CEISeR meta model, Import DSL meta model and local configuration
- Supports
  - Editing of job definition files (manifest, architecture, binding, infrastructure; search, edit, delete, insert, change, ...)
  - Analysis of job results (model diff, constraint violations)
  - Analysis of model dumps (offline snapshot of partial or full model)
  - Analysis of a diff between two model revisions (diff of two snapshots)
- Various wizards for creating CEISeR jobs and CEISeR modells

Xplor perspective



Xplor perspective - 2

- Xplor Configuration
  - job view, where all imported jobs are visualized
- Xplor structure
  - This view is for modelling the CEISeR artifacts, adding and deleting top-level elements and their child-elements
- Xplor Element
  - Details view for a selected element in the Xplor structure view
- Xplor Search
  - Search the active model for some model-elements (you can choose only valid model-elements)
- Xplor Search Results
  - Result view of a search

Future Prospects – CEISeR Evolution

- With our strategic partner “Software AG”* we will shift CEISeR to a standard product (CentraSite) to decrease/ avoid unnecessary internal development effort and therefore to safe money (in the long run).

- We will realize human-oriented workflows with the bpm suite “webMethods” from SAG regarding the processes working with CEISeR to formalize the process

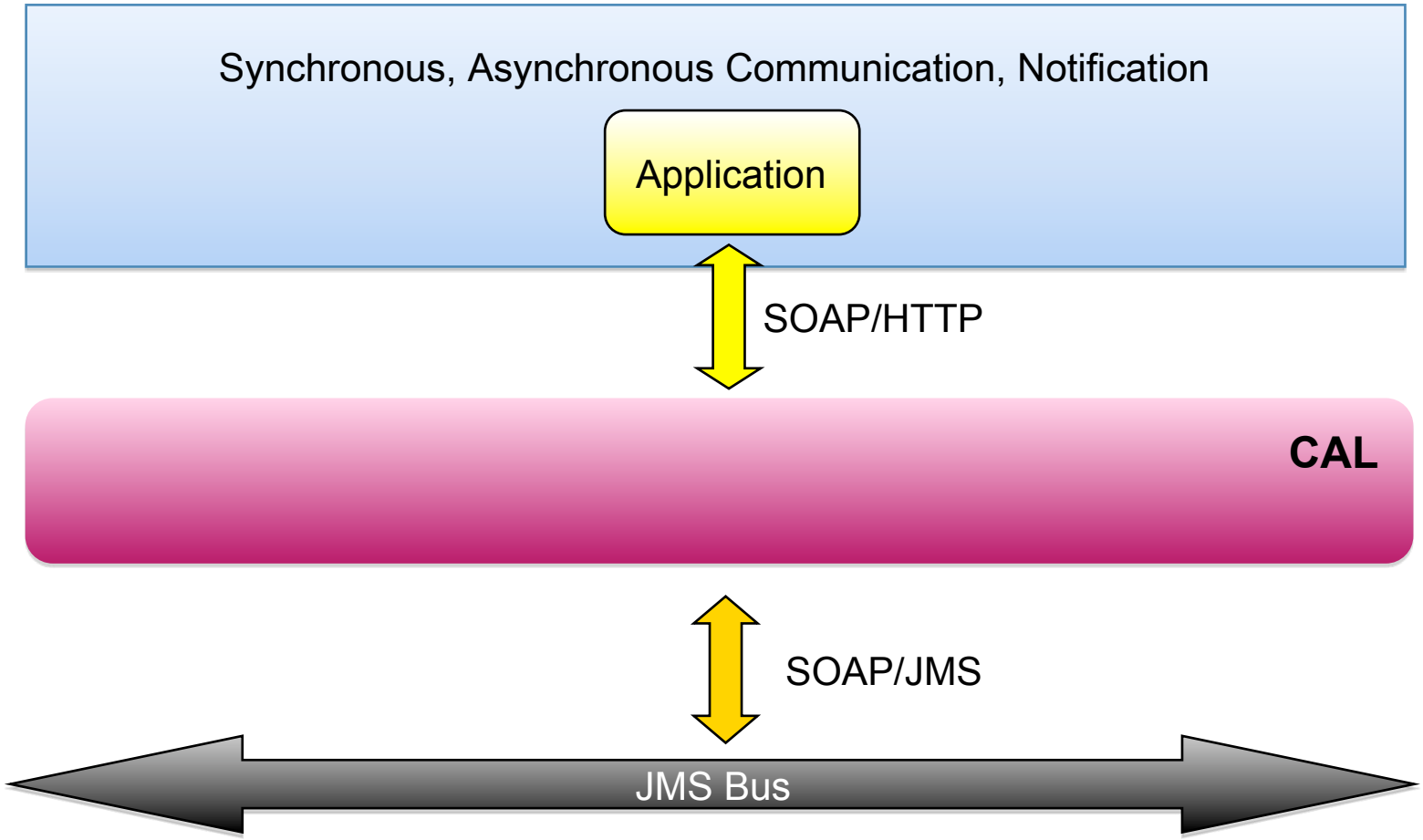
  - Defining a contract (service interface definition)
  - Defining an architecture, binding, ...
  - Service Lifecycle management

* http://www.computerwoche.de/knowledge_center/soa_bpm/1862456/

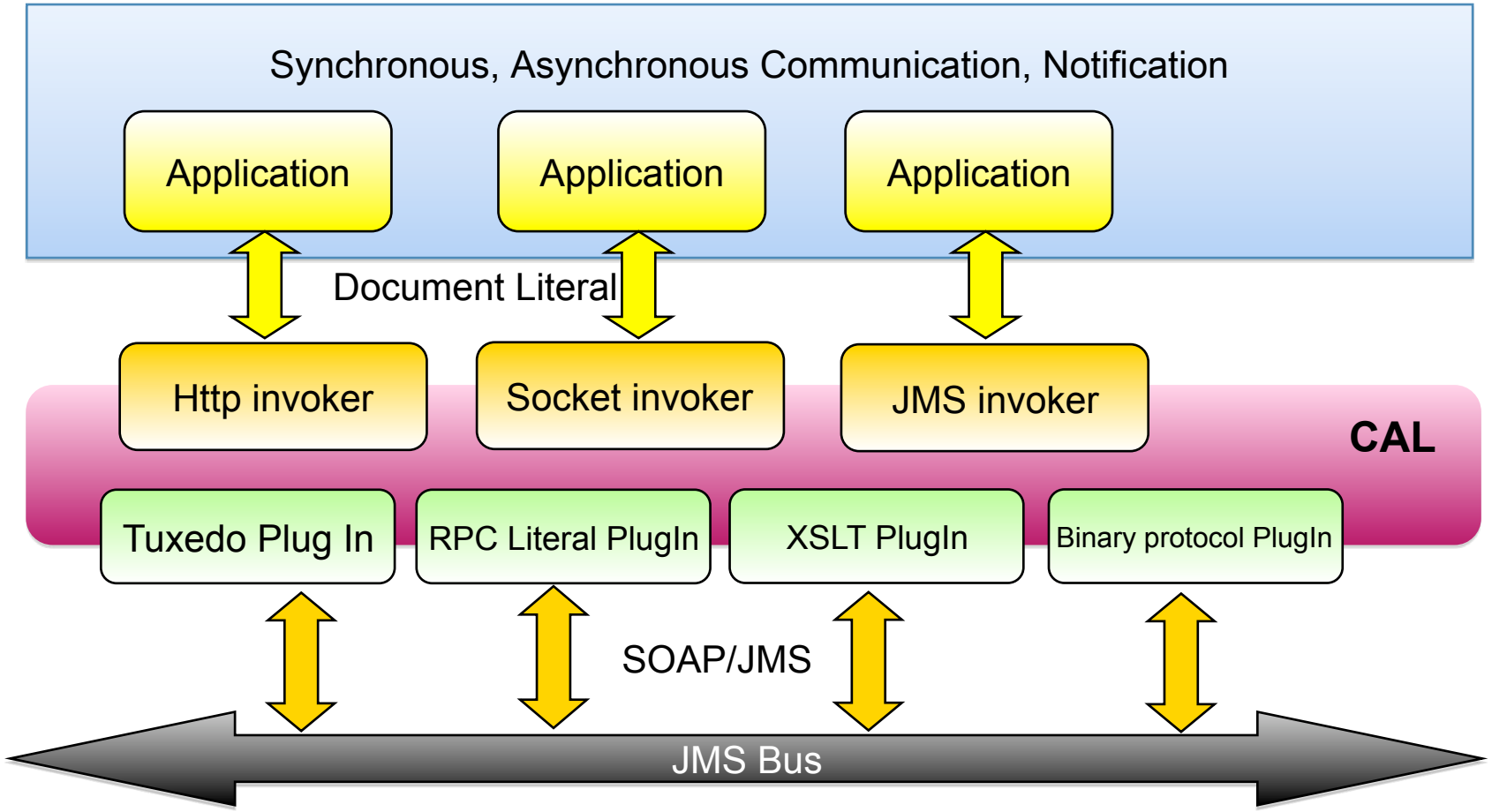
SOA Backplane - Runtime

Here is the action ... – Details of the ESB

The Enterprise Service Bus – in general



SOA Backplane – Common Access Layer (CAL) - ESB



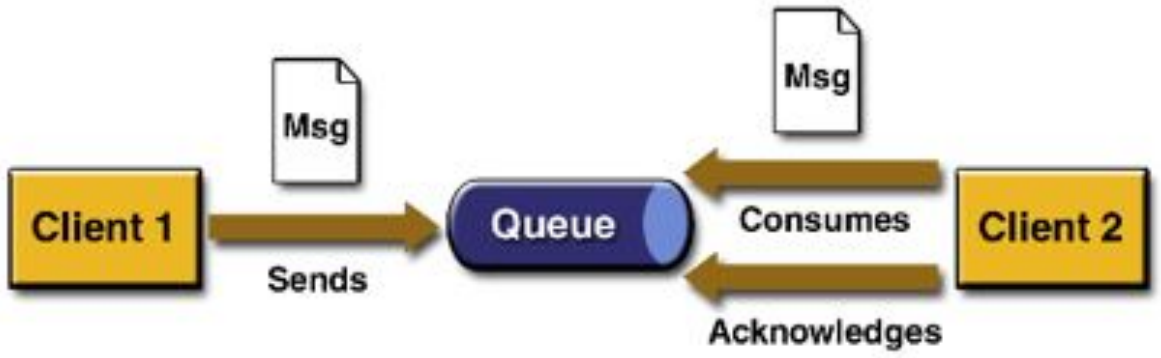
Existing Plugins/Adapter

- XSLT transformation (needs two XSLTs as configuration data)
  - Uses XSL transformation to map arbitrary XML messages
  - One example is the useage of SAP Netweaver (in TMAT)
- Tuxedo Plugin (needs the XSDs as configuration data)
  - Transforms flat FMLBuffer to structured data and vice versa
- Binary Plugin (needs a link to Java-library as configuration data)
  - Maps XML to a binary structure and vice versa
  - Used as base for Convergys Rater Plugin (MyFaves project)
- RPC literal Plugin (no config data needed)
  - Used for proprietary internal predecessor message format
  - Transforms RPC literal SOAP to document literal wrapped

Java Message Service – JMS

- JMS is the Java API for accessing MOM (message oriented middleware)

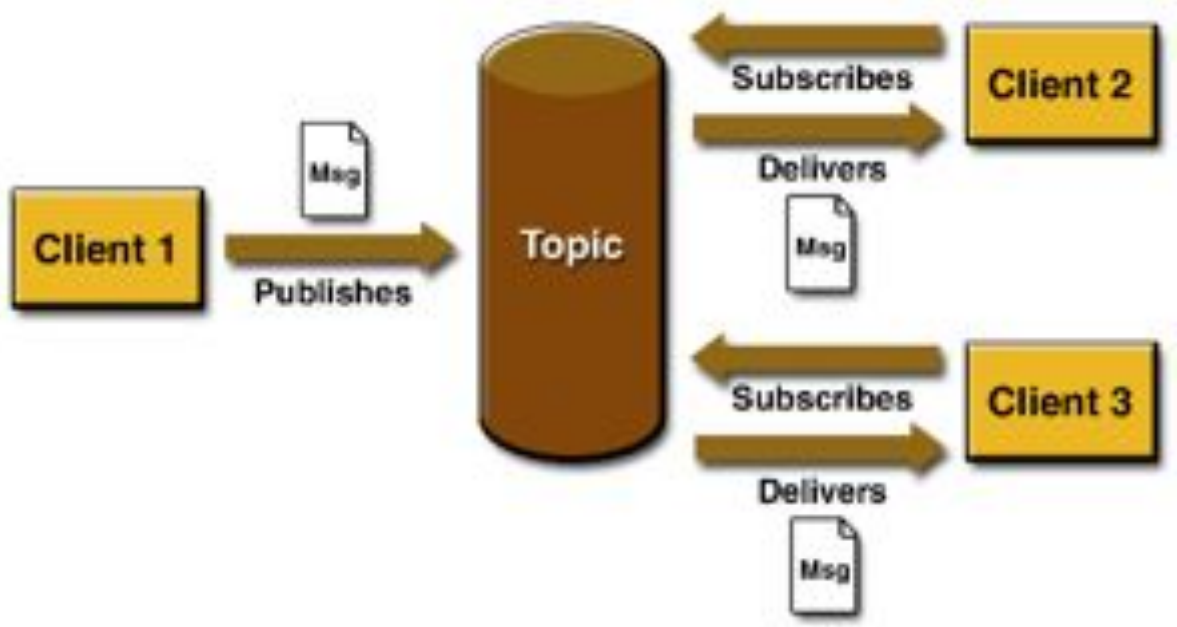
- A message is transported from a sender to a consumer via a queue asynchronously at least once, each message has only one consumer
- PTP:



- All participants must use JMS to take part in the message exchange

Java Message Service – JMS -2

- For publish/subscribe one can use a topic, thus there can be multiple consumers for one message
- This results in a timing dependency -> durable

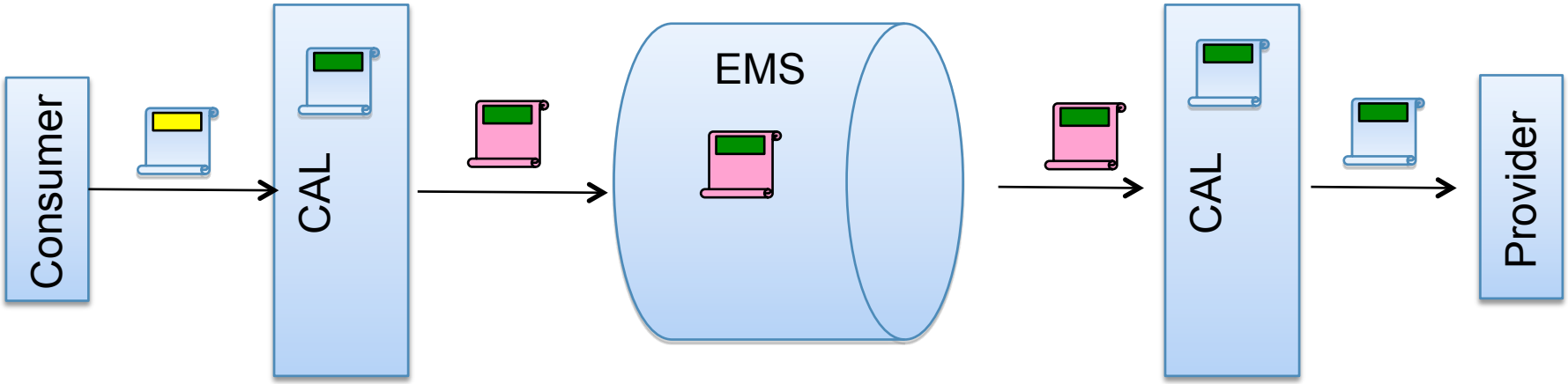


Java Message Service - Bus

- When using the Tibco EMS, different EMS servers can be connected to a hub and spoke
- Thus an international JMS Bus can be established using only a minimal set of interconnections
- A queue/topic can be accessed from a JMS client connected anywhere to the bus
- A globally accessible queue is residing on the central EMS server and a “remote-queue” is created as a proxy on each EMS server that will provide access to it

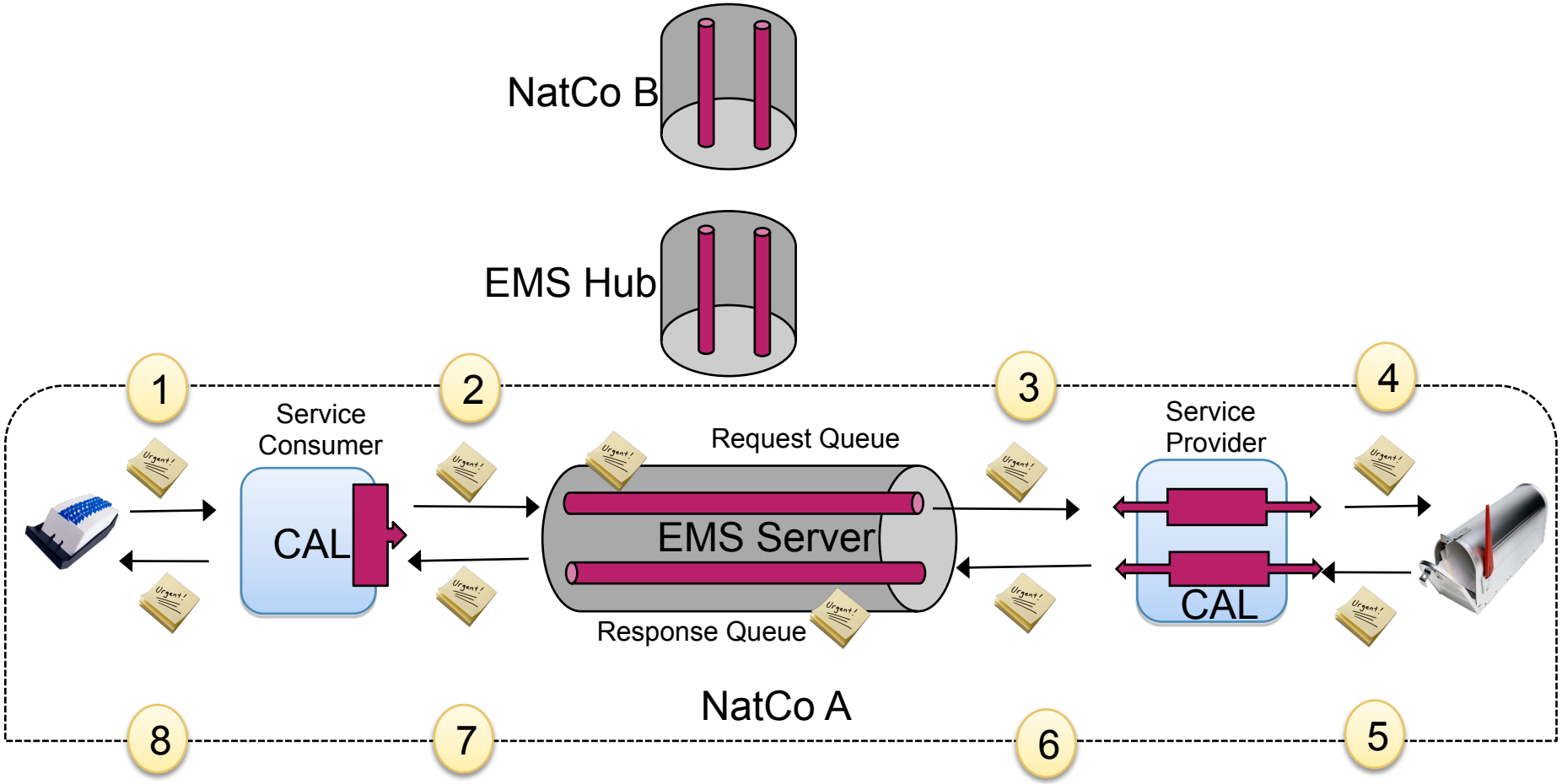
CAL – message flow* (synchronous request reply, consumer to provider view)

The CAL verifies only the EIMessage-Context of each message

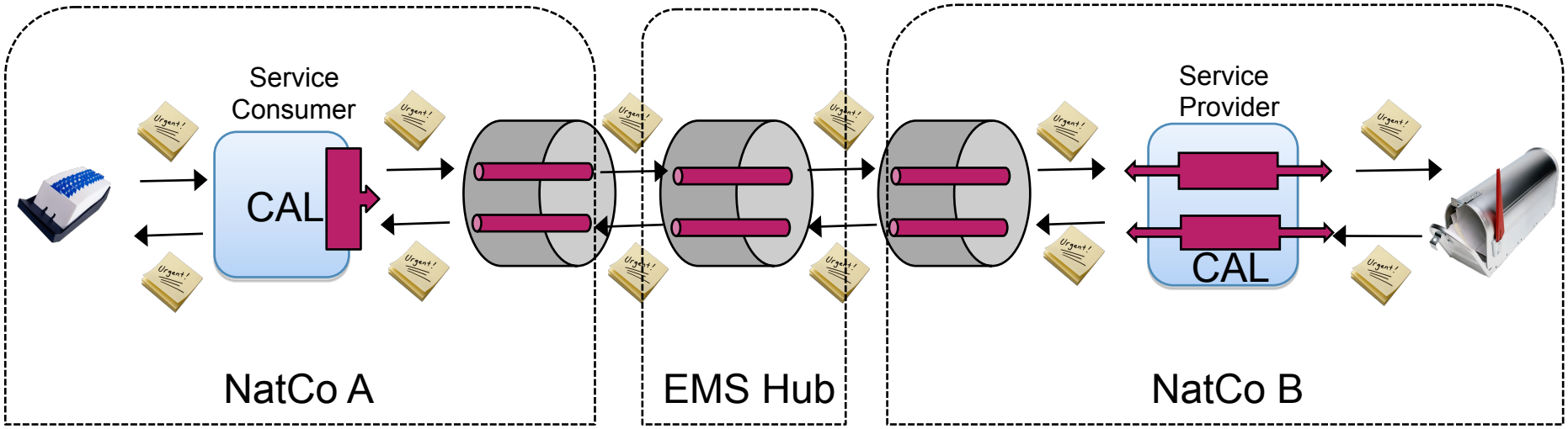


*data flow

Local message flow (only within one NatCo)



Global message flow (between two NatCos)



CAL log points – synchronous MEP - Example

Basic Service Invocation Scenario (Example: TMDE – TMNL)

1 – A ServiceConsumer sends a request to the CAL in TMDE ;

2 – The TMDE CAL delivers the request to the International Messaging Transport (EMS);

3 – The CAL in TMNL receives the request from the International Messaging Transport (EMS);

4 – The CAL in TMNL sends the request to the ServiceProvider ;

5 – The ServiceProvider sends the reply to the CAL in TMNL ;

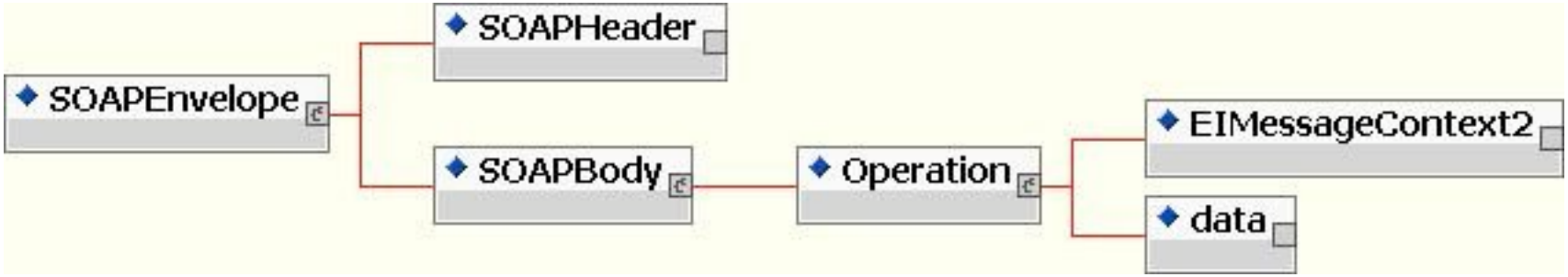
6 – the CAL in TMNL delivers the reply to the International Messaging Transport (EMS);

7 – The CAL in TMDE reads the reply from the International Messaging Transport (EMS);

8 – The CAL in TMDE sends the reply to the ServiceConsumer ;

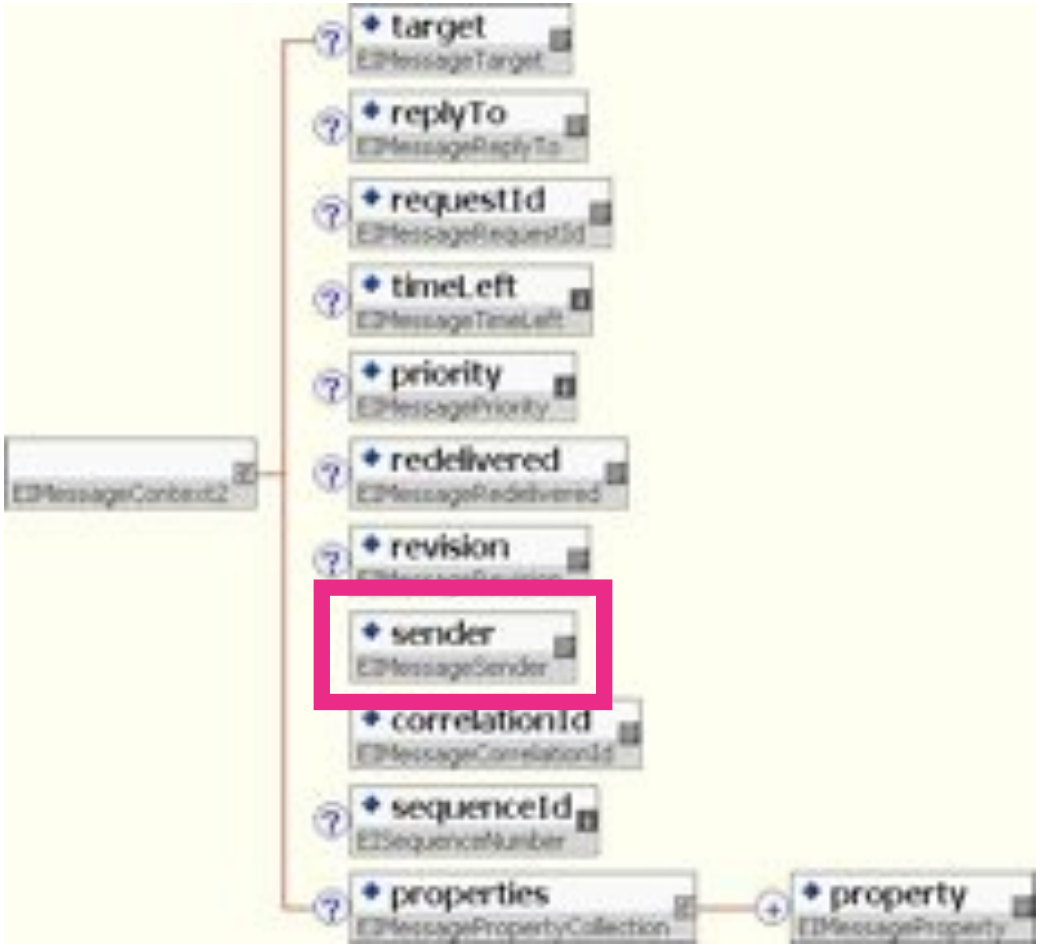
Message Structure

- SOABP conforming SOAP messages adhere to a common structure:
  - Due to the limitations of some web service toolkits SOA Backplane makes currently no use of SOAP header
  - Instead all SOABP relevant information is encapsulated in the eiMessageContext
  - Subsequent to the eiMessageContext follows the business relevant payload



eiMessageContext

- It carries the necessary information the CAL need to route the message (static routing)
- It enables business activity monitoring
- Message warehouse (LMS)
- Service policy enforcement



SOA Backplane – Provisioning time

Services from the assembly line

CAL runtime configuration by CEISeR and configuration agents

- The CAL runtime configuration consists of a set of WARs and EARs (JEE deployment artefacts)
- A WAR is containing the configuration for service consumer speaking to the CAL
- An EAR is containing the configuration for the CAL to call service providers
- Basically no code will deployed (except special customer plug-in code)
- By the usage of JEE deployment artefacts (WAR & EAR) the hot deployment mechanism of the CALs underlying JBoss can be utilized to achieve “hot configuration” for SOABP

How is the naming determined for queue names at the EMS server?

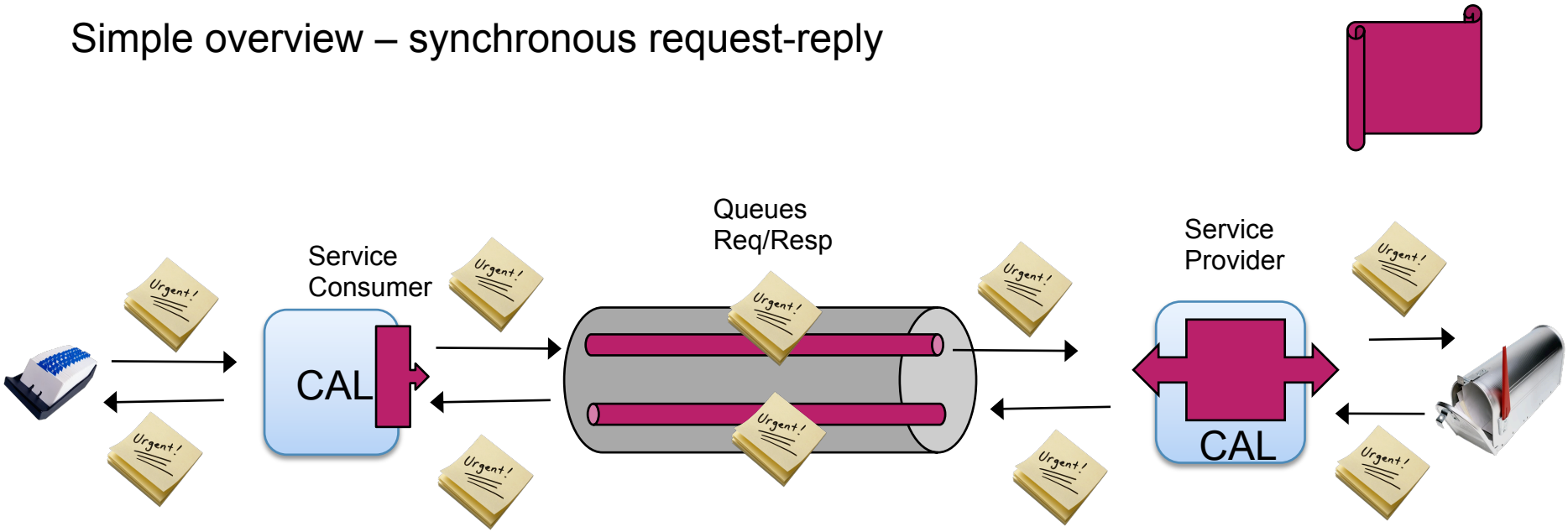
- It starts with the CQN (CEISeR qualified name) of the providing port component
- Suffix is .glob or .loc for global or local queues
- Suffix is .req or .resp for request or response queues

```
  tmo.ei.icc.training.architecture.AsynchProvider.PRV.activateAccount.glob.resp
  tmo.ei.icc.training.architecture.AsynchProvider.PRV.activateAccount.loc.req
  tmo.ei.icc.training.architecture.AsynchProvider.PRV.activateAccount.loc.resp
tcp://localhost:7222> _
```

Service Provision in a Nutshell

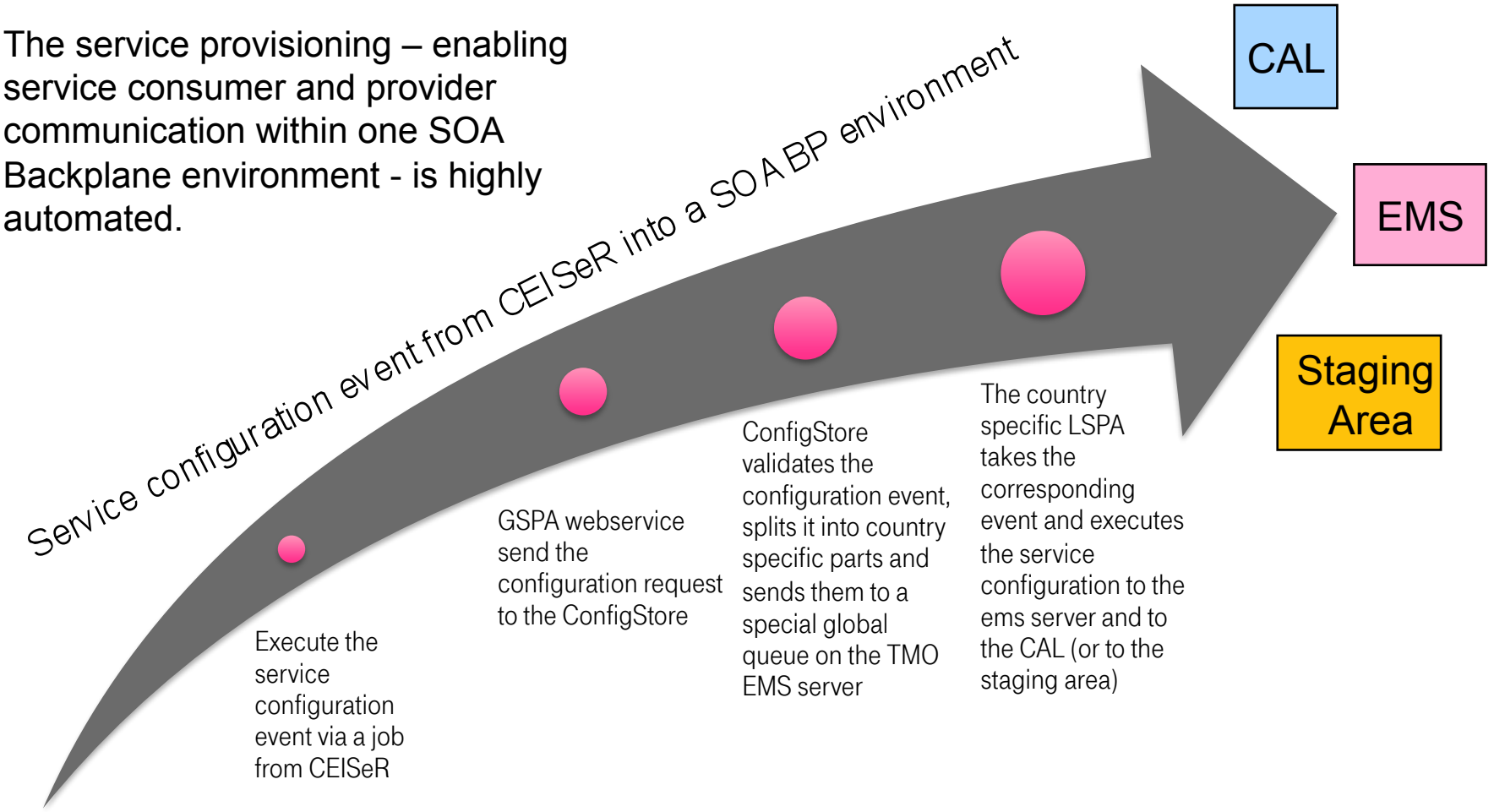
- A service provision enables the communication between service consumer and service provider within one specific SOA Backplane environment

Simple overview – synchronous request-reply



Service Provisioning – in general

The service provisioning – enabling service consumer and provider communication within one SOA Backplane environment - is highly automated.



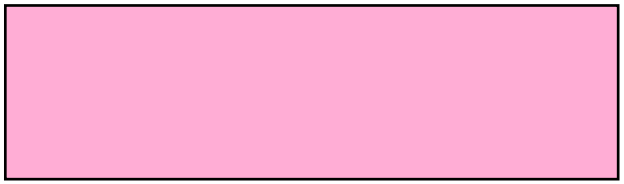
Legend for Service Provisioning

Service Configuration Event

*.ear files – provider configuration

CAL – Common Access Layer

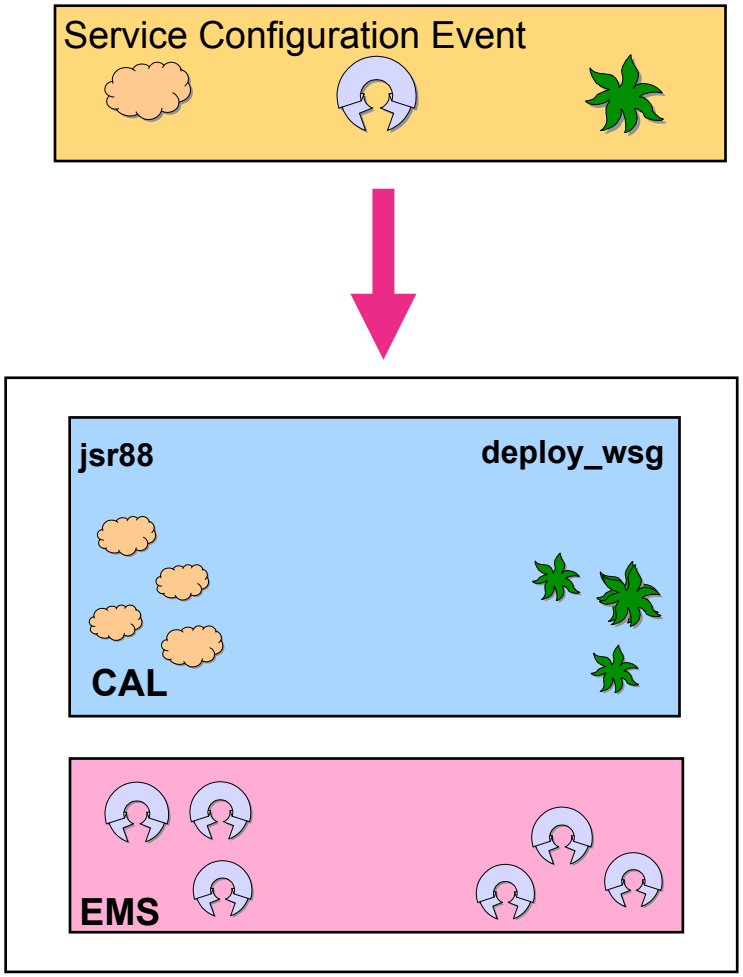
*.war files – user configuration



EMS Server

Queues for the EMS server

Service Provisioning– Deployment (new provider, new user)

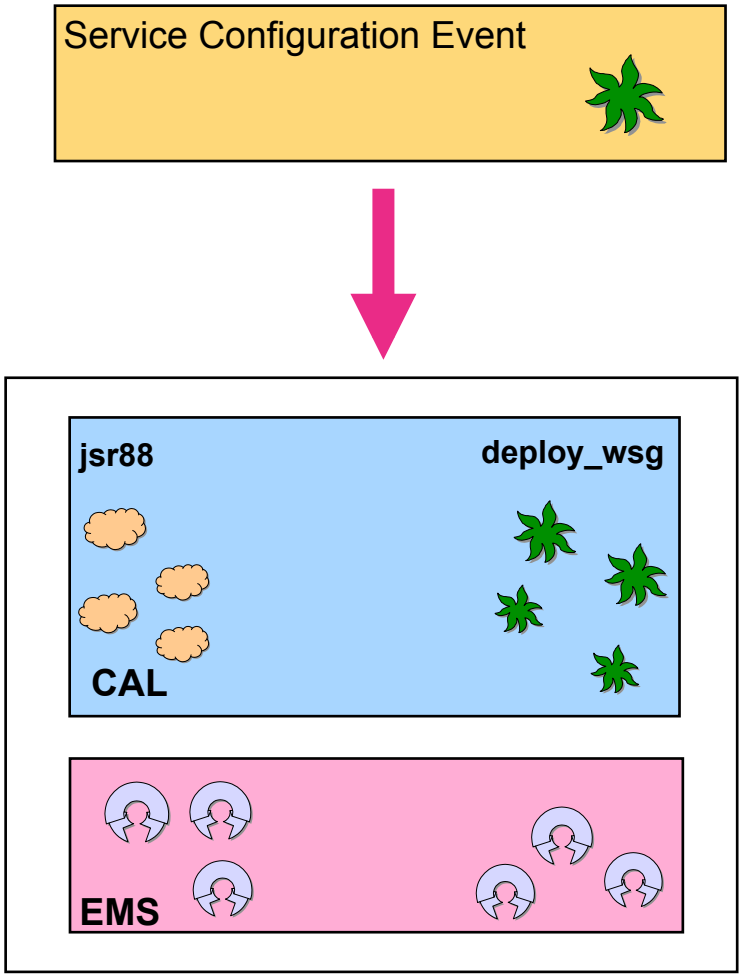


- New service provider
- New Service consumer for the new service provider
- Communication via JMS

Risks:

- No risks! Only new artefacts will be delivered into the CAL/ EMS

Service Provisioning– Redeployment (new user of an existing provider – redeployment of a war file)

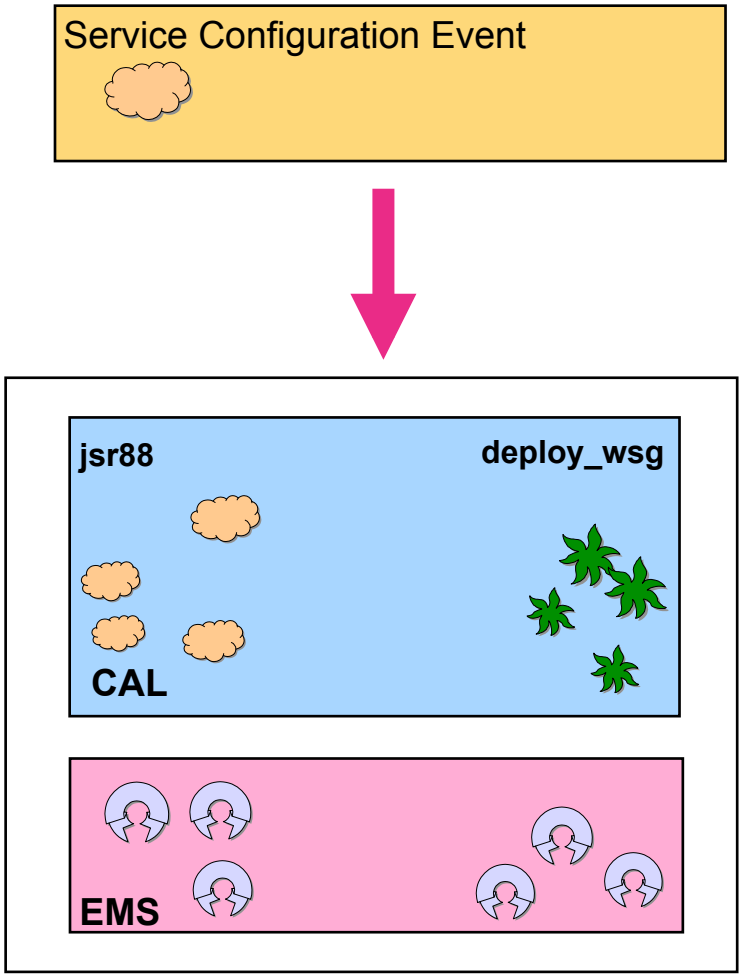


- additional service consumer for one service provider

Risks:

- Small risks!
- Only one war file will be changed within the CAL (the old user and the new one)

Service Provisioning– Redeployment (change of a providers' endpoint)

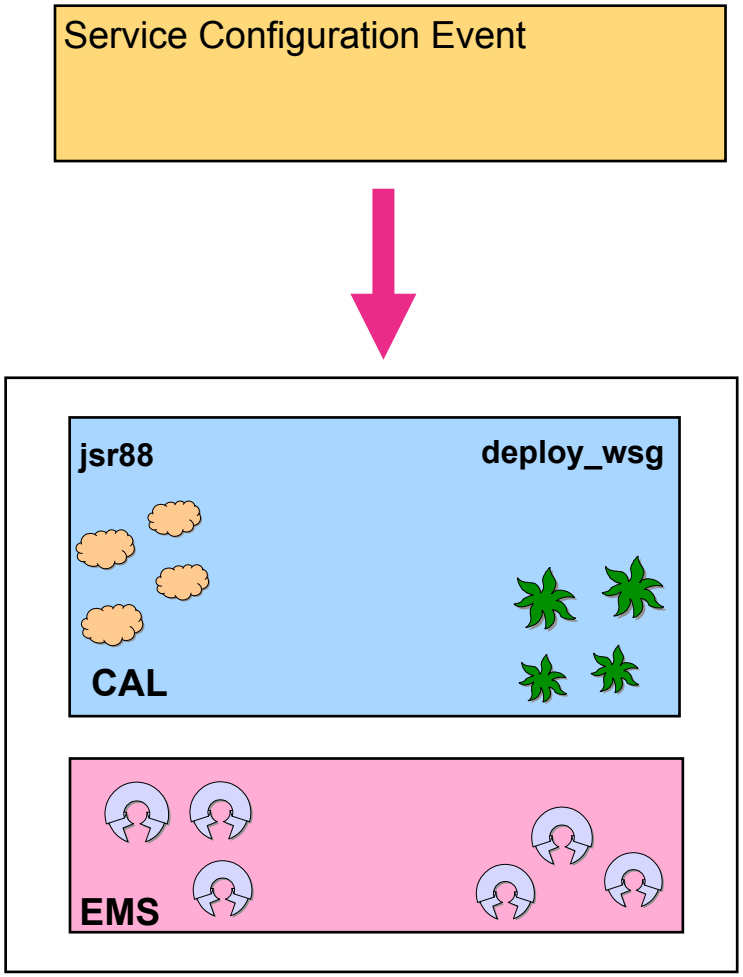


- Change of a endpoint of a service provider

Risks:

- Small risks
- Only one ear file will be changed

Service Provisioning– Undeployment (remove of a provider and his user)



- Remove of a service provider
- All belonging consumer will be removed too

Risks:

- No risks! Only the unneeded artefacts will be removed from the CAL
- The queues on the EMS server are untouched

Service Provisioning - Summary

- All changes within a service configuration affected only the service runtime configuration (either provider or consumer) from the requestor of the changes
- There are usually no risks and side effects for the whole service configuration (especially for the existing configuration which will be untouched)
- **All service provider changes** are encapsulated from the other provider configurations (EAR files)
- **Near all service consumer changes** are encapsulated from the other consumer configurations, except those changes as described before (new additional service user of a existing service provider) (WAR files)

CEISeR/ GCA/ ConfigStore/ LCA – Service Provisioning

- There are various ways to generate a service configuration event
- You can generate a service configuration event (artefacts) for
  - A whole environment (only useful in a development environment or a virtual environment)
  - BindingSet (most used way)
  - BindingComponentDispatcher (CAL instance of one NatCo)
  - ProvidingPortBinding
  - UsingPortBinding
  - All using and/or providing ports of a specific application
- It is possible to generate either DEPLOYMENT or UNDEPLOYMENT events
- There are special tasks in the job description file (manifest file)

Service Provisioning Event - validating

- 1 – before sending the SOAP request to the GCA
  - CEISeR validates against the XSD
- 2 – before splitting the config event
  - The GCA validates the number of expected service artefacts with the number of artefacts within the library file(s)
- 3 – before sending the splitted config event to the EMS server
  - The ConfigStore validates the CAL and EMS instances with the information from his own database, if some discrepancies occur, the ConfigStore aborts the delivery of the configuration event

ConfigStore

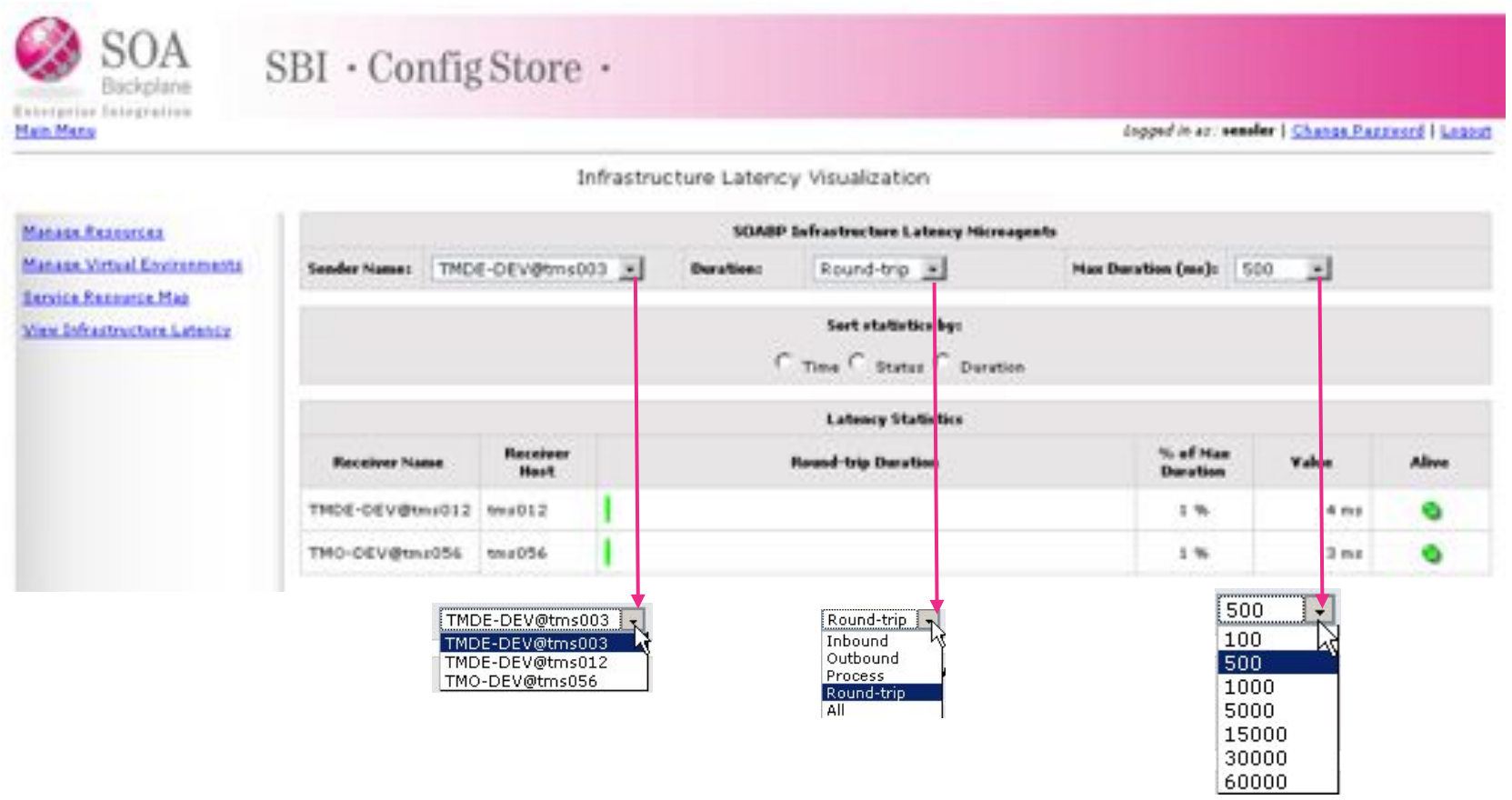
- Central component for verification of a service provision
- Monitors each provisioned artefact (services and queues)
- Auditing for who send what into which environment
- Monitoring of the runtime components
- Near future: automatic End-2-End testing to ensure that the service provider is available

ConfigStore – Manage Resources

ConfigStore – Manage Environments

ConfigStore – Service Resource Map

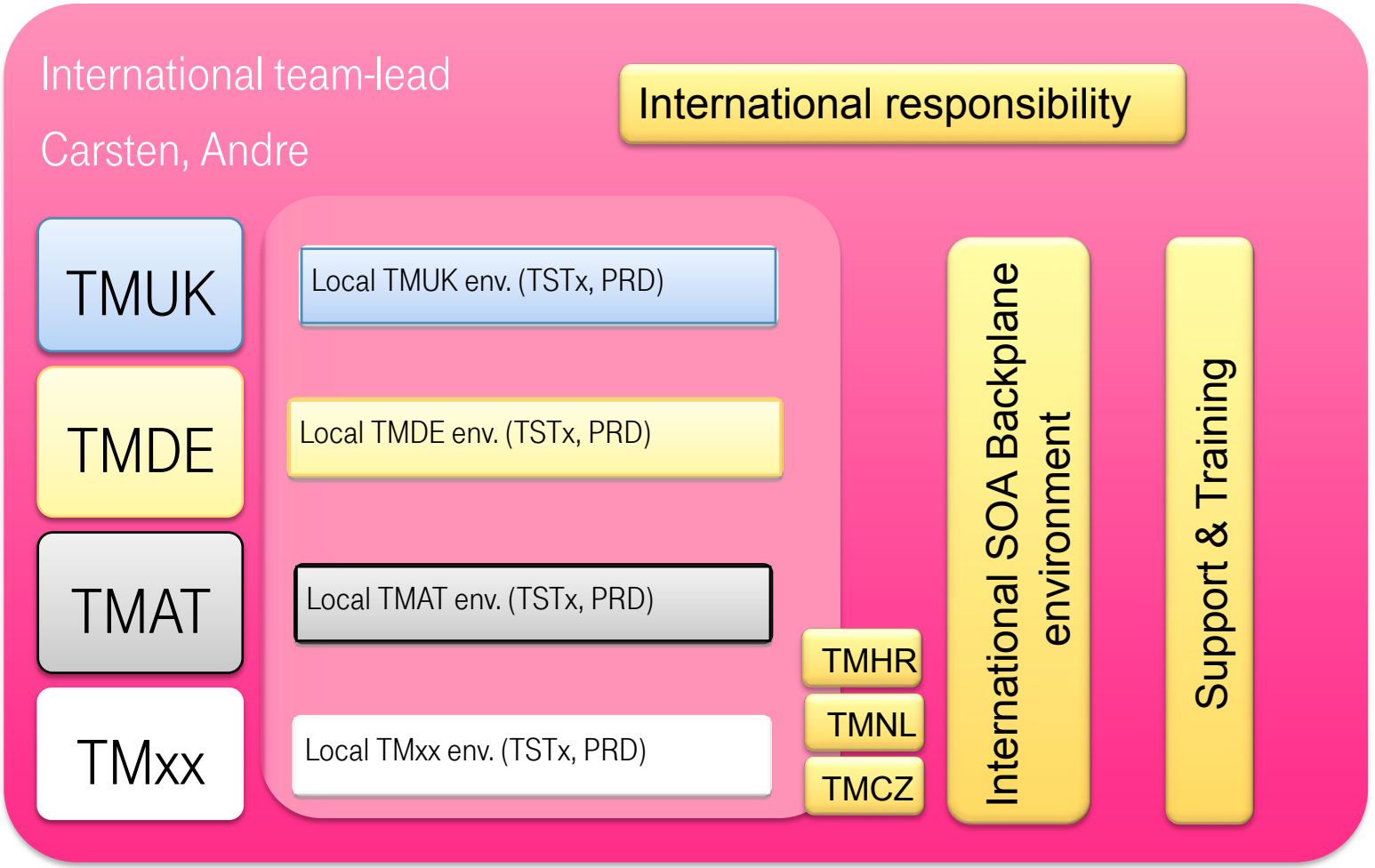
ConfigStore – View Infrastructure Latency



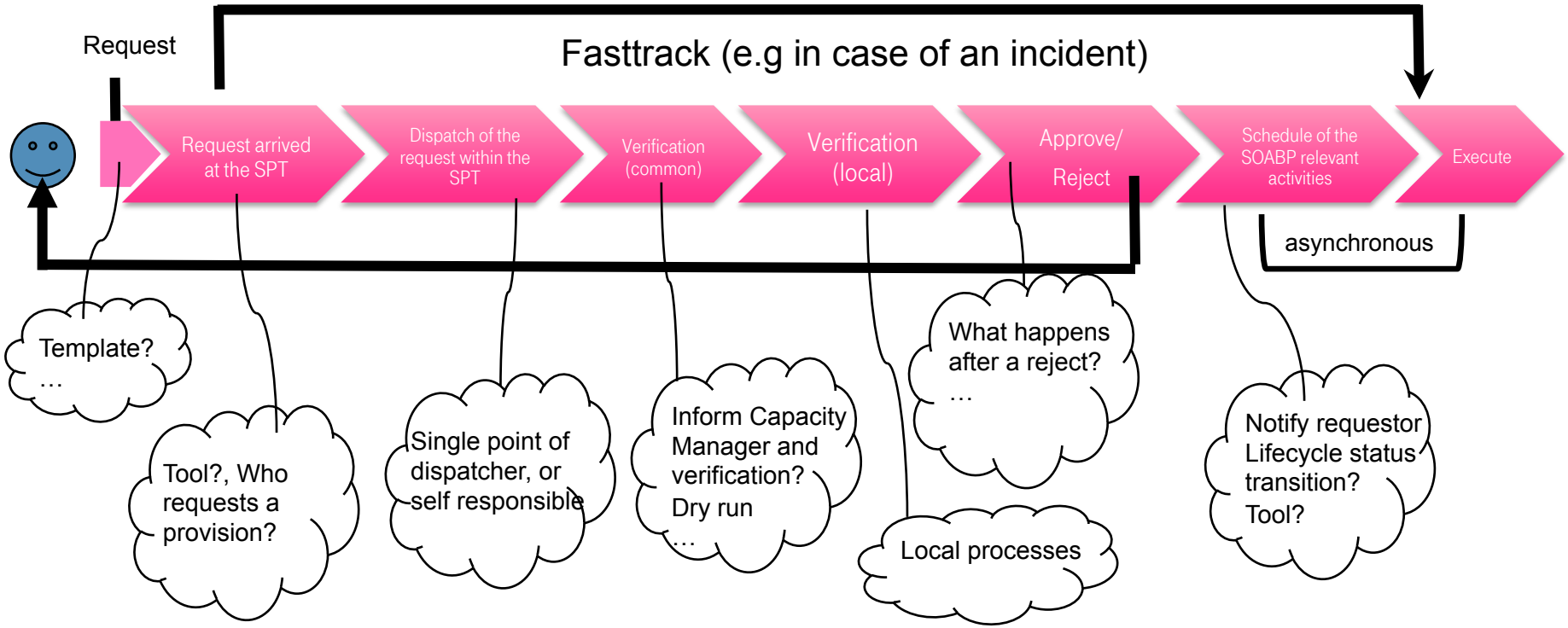
Organizational aspects – international focus

International Service Provisioning Team and Process

Service Provisioning Team - Structure



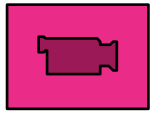
“Process” for requesting and executing a service provision (work in progress)



SPT = Service Provisioning Team

proposal

Summary



Enjoy....

Discussion

Thank you.

Questions?

Links

- Java Messaging Service
  - http://java.sun.com/products/jms/tutorial/1_3_1-fcs/doc/jms_tutorialTOC.html

- SOA@T-Mobile – vollautomatische Service Provisionierung auf dem ESB Teil 1-3, Javamagazin 10/08 – 12/08, C. Sensler & A. Karalus
  - http://www.sensler.de/data/JM_10.08_Sensler_TMobile1.pdf
  - http://www.sensler.de/data/JM_11.08_Sensler_TMobile2.pdf
  - http://www.sensler.de/data/JM_12.08_Sensler_TMobile3.pdf (folgt im Dezember)